Week 5 - Assignment

Programming for Data Science 2024

Exercises for the topics covered in the fifth lecture.

The exercise will be marked as passed if you get **at least 10/15** points.

Exercises must be handed in via **ILIAS** (Homework assignments). Deliver your submission as a compressed file (zip) containing one .py or .ipynb file with all exercises. The name of both the .zip and the .py/.ipynb file **must** be *SurnameName* of the two members of the group. Example: Riccardo Cusinato + Athina Tzovara = *CusinatoRiccardo_TzovaraAthina.zip* .

It's important to use comments to explain your code and show that you're able to take ownership of the exercises and discuss them.

You are not expected to collaborate outside of the group on exercises and submitting other groups' code as your own will result in 0 points.

For questions contact: *riccardo.cusinato@unibe.ch* with the subject: *Programming for Data Science 2024*.

Deadline: 14:00, March 28, 2024.

Exercise 1 - Fitbit dataset 3 points

We will work with three datasets - 'activity.csv', 'calories.csv', and 'last_participant.csv', which contains activity tracker data from https://www.kaggle.com/datasets/arashnic/fitbit

If you are unable to do this exercise, you can load the dataset 'combined_solution.csv' for the next exercise.

1. **Data preparation** (*1 point*)

   - Load the two datasets 'activity.csv' and 'calories.csv'.
   - Use pd.to_datetime to standardize the ActivityDate columns (https://pandas.pydata.org/docs/reference/api/pandas.to_datetime.html)

In [ ]:
```
###
# YOUR CODE HERE
###
```

2. **Merging** (*1 point*)

   - Consider what information is shared between the two datasets and merge them. Keep in mind that the order of rows is not the same in both datasets!
   - Print out the mean "TotalSteps" of the merged DataFrame at this point.

In [ ]:
```
###
# YOUR CODE HERE
###
```

3. **Concatenation** (*1 point*)

- The data of one additional participant exists in 'last_participant.csv'. Load this dataset and concatenate it with the merged dataset generated above
- Print out the mean "TotalSteps" again

```
In [ ]: ###
        # YOUR CODE HERE
        ###
```

Exercise 2 - Working with missing data 5 points

In our dataset, some values are missing from the 'TotalSteps' and 'Calories' columns.

We can try to approximate these missing values with the data we got.

You can load the dataset 'combined_solution.csv' if you were unable to complete the previous exercise.

1. **Filling in missing values** (*3 points*)

   - Calculate the mean steps per calory burnt and mean calories burnt per step, by averaging across all observations in the dataset and then computing the ratio. Print out both values.
   - Fill in the null values in the columns 'Calories' and 'TotalSteps' where possible. To fill the values you have to use the factors *"TotalSteps/Calories"* and *"Calories/TotalSteps"* calculated in the previous point, using one of the two information to fill the other.
   - Print out the mean of the columns 'TotalSteps' and 'Calories' before and after filling the missing values.

```
In [ ]: ###
        # YOUR CODE HERE
        ###
```

2. **Dropping missing values** (*2 points*)

   - Print how many null values there are in the 'Calories' and 'TotalSteps' columns, respectively.
   - Drop the rows where **both** 'Calories' and 'TotalSteps' are missing.
   - Print number of rows in the final dataset.

```
In [ ]: ###
        # YOUR CODE HERE
        ###
```

Exercise 3 - Multi-index 7 points

In this exercise you will create and manipulate a multi-index dataframe. First, let's create the dataframe for the exercise:

```
In [ ]: import pandas as pd

        df = pd.DataFrame(
            {
                "idx": [0, 1, 2],
                "A_X": [1.1, 1.1, 1.1],
                "A_Y": [1.2, 1.2, 1.2],
        "B_X": [1.11, 1.11, 1.11],
        "B_Y": [1.22, 1.22, 1.22],
    }
)
```

1. Set the column *idx* as the index of the dataframe. (*1 point*)

In [ ]:
```
###
# YOUR CODE HERE
###
```

2. Create a multi-column stucture. (*3 points*)
    - Set the columns *A, B* on the first level and *X, Y* on the second level, taken from the combinations in the original dataframe.
    - Set the names of the two new levels as "L1" and "L2", respectively.
    - Print the resulting dataframe.

In [ ]:
```
###
# YOUR CODE HERE
###
```

3. From the previous dataframe, re-create a dataframe with a single column level. (*3 points*)
    - Create a new column from the first level (L1) of the multi-column. At this point your columns should be ['L1', 'X', 'Y'], with name 'L2'. **NB** The DataFrame method *reset_index* is useful for this part.
    - Rename the newly-created column as "letter" and the name of the column level as "L". Use the appropiate pandas methods for this.
    - Print the resulting dataframe.

In [ ]:
```
###
# YOUR CODE HERE
###
```